Iterative Kompression

Cluster Vertex Deletion (CVD)

Problemstellung

Gegeben einen Graphen, lösche möglichst wenige Knoten, sodass der resultierende Graph ein Cluster-Graph ist. Ein Cluster-Graph ist ein Graph bei dem jede Zusammenhangskomponente eine Clique ist.

Ziel: Nutze iterative Kompression, um einen FPT-Algorithmus zu erhalten. Reduziere CVD dazu zunächst auf CVD-Compression und reduziere davon weiter auf Disjoint-CVD. Löse dann Disjoint-CVD.

Schritt 1: CVD-Compression

CVD-Compression: Gegeben einen Graphen $G = (V, E)$ zusammen mit einer Lösung $Z \subseteq V$ der Größe $k + 1$, gibt es eine Lösung der Größe k?

Man kann sich zunächst klar machen, dass ein FPT-Algo für CVD-Compression auch einen FPT-Algo für CVD liefert:

- Wir bauen den Graph iterativ auf, indem wir nach und nach Knoten hinzufügen.
- Jeder neue Knoten wird zunächst in die Lösung aufgenommen → neue Lösung die ggf. um 1 zu groß ist.
- Kompressions-Algo macht die Lösung wieder kleiner oder entscheidet, dass es keine Lösung gibt. In letzterem Fall gibt es auch für gesamt G keine Lösung.

Schritt 2: Disjoint-CVD

Disjoint-CVD: Gegeben einen Graphen $G = (V, E)$ zusammen mit einer Lösung $Y \subseteq V$ der Größe $k + 1$, gibt es eine zu Y disjunkte Lösung der Größe k?

Auch hier kann man sich wieder klar machen, dass ein FPT-Algo für Disjoint-CVD einen FPT-Algo für CVD-Compression liefert: Möchte man CVD-Compression lösen, dann iteriert man über die Teilmengen $Y \subseteq Z$ und sucht eine Lösung, die zu Y disjunkt ist, aber $Z \setminus Y$ komplett enthält. Dann kann man $Z \setminus Y$ gleich löschen und ist beim Disjoint-CVD Problem.

Schritt 3: Reduktionen und strukturelle Eigenschaften der Instanz

Betrachte eine Instanz von Disjoint-CVD und sei $X = V \setminus Y$. Wir können folgende Beobachtungen machen:

- Da Y eine Lösung ist muss $G[X]$ ein Cluster-Graph sein.

- Da wir einen Cluster-Graphen erhalten wollen, indem wir nur Knoten aus X löschen, muss $G[Y]$ schon ein Cluster-Graph sein.

Sowohl Y als auch X bilden also eine Menge von Cliquen. Wenn wir jetzt einen Knoten $v \in X$ betrachten, dann können wir folgende Beobachtungen machen (man findet sonst jeweils leicht einen P_3):

- Wenn v Kanten zu zwei oder mehr unterschiedlichen Cliquen in Y hat, so müssen wir v löschen.
- Wenn v Kanten zu einem Knoten in einer Clique aus Y hat, so muss v Kanten zu allen Knoten aus dieser Clique haben.

Die beide Reduktionsregeln können wir erschöpfend anwenden (und dabei k entsprechend reduzieren). Damit hat dann jeder Knoten in X entweder keine Kanten zu Y oder zu allen Knoten genau einer Clique aus $G[Y]$.

Schritt 4: Lösung

Betrachte eine Clique C in $G[X]$ und seine u und v zwei Knoten aus C. Wenn u und v zu unterschiedlichen Cliquen in Y verbunden sind (oder einer keine Kanten zu Y hat), dann darf nur einer der beiden Knoten behalten werden. Das heißt, für C wählen wir eine Clique aus $G[Y]$ aus und können dann alle die Knoten behalten, die Kanten in diese Clique haben (alternativ können wir auch keine Clique auswählen und alle Knoten behalten, die keine Kanten in Y haben). Beachte, dass jede Clique in $G[Y]$ auf diese Art nur einmal ausgewählt werden darf, da wir sonst wieder einen P_3 erhalten.

Damit müssen wir nur noch ein gewichtsmaximales Matching in einem bipartiten Graphen finden.

Split Vertex Deletion (SVD)

Problemstellung

Gegeben einen Graphen, lösche möglichst wenige Knoten, sodass der resultierende Graph ein Split-Graph ist. Ein Split-Graph ist ein Graph, dessen Knoten in zwei Mengen partitioniert werden können, sodass die eine Menge eine Clique und die andere eine unabhängige Menge (Menge isolierter Knoten) induziert. Eine solche Aufteilung nenne wir auch *Split*.

Ziel: Löse **Disjoint-SVD**, also die Problemvariante, bei der man zusätzlich zum Graphen $G = (V, E)$ und dem Parameter k auch eine Lösung $Y \subseteq V$ der Größe $k + 1$ gegeben hat. Sei außerdem im Folgenden $X = V \setminus Y$.

Strukturelle Einsicht: wenige relevante Aufspaltungen

Da Y eine Lösung ist und wir gleich abbrechen können, wenn ganz X keine Lösung ist, können wir annehmen dass sowohl $G[Y]$ als auch $G[X]$ Split-Graphen sind.

Mal angenommen, wir hätten alle Knoten schon eine Aufspaltung in zwei Mengen gegeben, sodass diese Aufspaltung im finalen Split-Graphen ein Split ist. Dann ist die Hoffnung, dass das Problem dadurch deutlich einfacher wird. Ziel ist es im Folgenden erstmal einzusehen, dass es nur polynomiell viele solcher Aufspaltungen gibt. Danach schauen wir dann, wie wir das Problem für eine gegebene Aufspaltung lösen können.

Betrachte irgendeinen Split (A, B) für $G[X]$ (d.h. $A, B \subseteq X$ mit $A \cap B = \emptyset$ und $A \cup B = X$), wobei $G[A]$ die Clique und $G[B]$ die unabhängige Menge ist. Betrachte außerdem zwei Knoten $u, v \in A$. Falls u und v beide nicht gelöscht werden, dann können sie im Split des finalen Split-Graphen nicht beide in der unabhängigen Menge enthalten sein (sie sind ja mit einer Kante verbunden). Die analoge Beobachtung gilt umgekehrt auch für den Fall, dass $u, v \in B$. Daraus folgt, dass sich der finale Split von der durch (A, B) induzierten Aufspaltung nicht stark unterscheiden kann. (Beachte aber, dass die von (A, B) induzierte Aufspaltung nicht unbedingt ein Split im finalen Graphen ist, da das ggf. nicht zu Y passt.)

Betrachte nun die folgende Menge von Aufspaltungen von X: Starte mit einem beliebigen Split (A, B) von $G[X]$. Betrachte alle Aufspaltungen (A', B') von X in zwei disjunkte Mengen, die sich von (A, B) erhalten lassen, indem bis zu ein Knoten von A zu B und bis zu einer von B zu A bewegt wurde. Beachte, dass dabei (A', B') nicht in jedem Fall ein Split ist. Wie oben argumentiert gilt aber, dass jeder Split von dem finalen Split-Graphen zu einer dieser Aufspaltungen (A', B') passt.

Beachte, dass die Anzahl Aufspaltungen, die man so erhalten kann, quadratisch ist. Mit einem ähnlichen Argument gibt es auch nur quadratisch viele Aufspaltungen von Y (tatsächlich sind es hier sogar weniger, da wir von Y nichts löschen dürfen und daher nur Splits von $G[Y]$ betrachtet werden müssen). Insgesamt können wir aber festhalten, dass es für V nur polynomiell viele Möglichkeiten der Aufspaltung gibt, die wir betrachten müssen. Wir können im Folgenden also davon ausgehen, eine feste Aufspaltung gewählt zu haben, die auf die oben beschriebene Art entstanden ist.

Algorithmus für feste Aufspaltung

Betrachte eine feste Aufspaltung (A, B) von G, wobei A die Clique und B die unabhängige Menge werden soll. Da wir Y komplett behalten wollen, müssen wir alle Knoten aus A löschen, die nicht mit allen Knoten aus $Y \cap A$ verbunden sind. Genauso müssen wir alle Knoten aus B löschen, die eine Kante zu einem Knoten aus $Y \cap B$ haben.

Danach müssen wir nur noch eine möglichst große Clique in $X \cap A$ und eine möglichst große unabhängige Menge in $X \cap B$ finden. Das geht in diesem Fall leicht, da wir wissen, dass $X \cap A$ schon fast eine Clique ist: die betrachtete Aufspaltung ist ja dadurch entstanden, dass wir einen Split für $G[X]$ hatten und höchstens einen Knoten von der unabhängigen Menge zu der Clique verschoben haben. Es gilt also, dass wir entweder schon eine Clique haben und nichts löschen müssen oder dass wir einen Knoten löschen müssen. Das lässt sich leicht testen. Analoges gilt für die andere Seite.